

Algorithms for Testing Monomials in Multivariate Polynomials

Zhixiang Chen Bin Fu Yang Liu Robert Schweller

Department of Computer Science
University of Texas-Pan American
Edinburg, TX 78539, USA
{chen, binfu, yliu, schweller}@cs.panam.edu

Abstract

This paper is our second step towards developing a theory of testing monomials in multivariate polynomials. The central question is to ask whether a polynomial represented by an arithmetic circuit has some types of monomials in its sum-product expansion. The complexity aspects of this problem and its variants have been investigated in our first paper by Chen and Fu (2010), laying a foundation for further study. In this paper, we present two pairs of algorithms. First, we prove that there is a randomized $O^*(p^k)$ time algorithm for testing p -monomials in an n -variate polynomial of degree k represented by an arithmetic circuit, while a deterministic $O^*(6.4^k + p^k)$ time algorithm is devised when the circuit is a formula, here p is a given prime number. Second, we present a deterministic $O^*(2^k)$ time algorithm for testing multilinear monomials in $\Pi_m \Sigma_2 \Pi_t \times \Pi_k \Pi_3$ polynomials, while a randomized $O^*(1.5^k)$ algorithm is given for these polynomials. The first algorithm extends the recent work by Koutis (2008) and Williams (2009) on testing multilinear monomials. Group algebra is exploited in the algorithm designs, in corporation with the randomized polynomial identity testing over a finite field by Agrawal and Biswas (2003), the deterministic noncommunicative polynomial identity testing by Raz and Shpilka (2005) and the perfect hashing functions by Chen *et al.* (2007). Finally, we prove that testing some special types of multilinear monomial is W[1]-hard, giving evidence that testing for specific monomials is not fixed-parameter tractable.

1 Introduction

1.1 Overview

We begin with two examples to exhibit the motivation and necessity of the study about the monomial testing problem for multivariate polynomials. The first is about testing a k -path in any given undirected graph $G = (V, E)$ with $|V| = n$, and the second is about the satisfiability problem. Throughout this paper, polynomials refer to those with multiple variables.

For any fixed integer $c \geq 1$, for each vertex $v_i \in V$, define a polynomial $p_{k,i}$ as follows:

$$\begin{aligned} p_{1,i} &= x_i^c, \\ p_{k+1,i} &= x_i^c \left(\sum_{(v_i, v_j) \in E} p_{k,j} \right), \quad k > 1. \end{aligned}$$

We define a polynomial for G as

$$p(G, k) = \sum_{i=1}^n p_{k,i}.$$

Obviously, $p(G, k)$ can be represented by an arithmetic circuit. It is easy to see that the graph G has a k -path $v_{i_1} \cdots v_{i_k}$ iff $p(G, k)$ has a monomial $x_{i_1}^c \cdots x_{i_k}^c$ of degree ck in its sum-product expansion. G has a Hamiltonian path iff $p(G, n)$ has the monomial $x_1^c \cdots x_n^c$ of degree cn in its sum-product expansion. One can also see that a path with some loop can be characterized by a monomial as well. Those observations show that testing monomials in polynomials is closely related to solving k -path, Hamiltonian path and other problems about graphs. When $c = 1$, $x_{i_1} \cdots x_{i_k}$ is multilinear. The problem of testing multilinear monomials has recently been exploited by Koutis [15] and Williams [22] to design innovative randomized parameterized algorithms for the k -path problem.

Now, consider any CNF formula $f = f_1 \wedge \cdots \wedge f_m$, a conjunction of m clauses with each clause f_i being a disjunction of some variables or negated ones. We may view conjunction as multiplication and disjunction as addition, so f looks like a "polynomial", denoted by $p(f)$. $p(f)$ has a much simpler $\Pi\Sigma$ representation, as will be defined in the next section, than general arithmetic circuits. Each "monomial" $\pi = \pi_1 \cdots \pi_m$ in the sum-product expansion of $p(f)$ has a literal π_i from the clause f_i . Notice that a boolean variable $x \in Z_2$ has two properties of $x^2 = x$ and $x\bar{x} = 0$. If we could realize these properties for $p(f)$ without unfolding it into its sum-product, then $p(f)$ would be a "real polynomial" with two characteristics: (1) If f is satisfiable then $p(f)$ has a multilinear monomial, and (2) if f is not satisfiable then $p(f)$ is identical to zero. These would give us two approaches towards testing the satisfiability of f . The first is to test multilinear monomials in $p(f)$, while the second is to test the zero identity of $p(f)$. However, the task of realizing these two properties with some algebra to help transform f into a needed polynomial $p(f)$ seems, if not

impossible, not easy. Techniques like arithmetization in Shamir [21] may not be suitable in this situation. In many cases, we would like to move from Z_2 to some larger algebra so that we can enjoy more freedom to use techniques that may not be available when the domain is too constrained. The algebraic approach within $Z_2[Z_2^k]$ in Koutis [15] and Williams [22] is one example along the above line. It was proved in Bshouty *et al.* [6] that extensions of DNF formulas over Z_2^n to Z_N -DNF formulas over the ring Z_N^n are learnable by a randomized algorithm with equivalence queries, when N is large enough. This is possible because a larger domain may allow more room to utilize randomization.

There has been a long history in theoretical computer science with heavy involvement of studies and applications of polynomials. Most notably, low degree polynomial testing/representing and polynomial identity testing have played invaluable roles in many major breakthroughs in complexity theory. For example, low degree polynomial testing is involved in the proof of the PCP Theorem, the cornerstone of the theory of computational hardness of approximation and the culmination of a long line of research on IP and PCP (see, Arora *et al.* [3] and Feige *et al.* [11]). Polynomial identity testing has been extensively studied due to its role in various aspects of theoretical computer science (see, for examples, Chen and Kao [9], Kabanets and Impagliazzo [13]) and its applications in various fundamental results such as Shamir's IP=PSPACE [21] and the AKS Primality Testing [2]. Low degree polynomial representing [16] has been sought for so as to prove important results in circuit complexity, complexity class separation and subexponential time learning of boolean functions (see, for examples, Beigel [5], Fu[12], and Klivans and Servedio [14]). These are just a few examples. A survey of the related literature is certainly beyond the scope of this paper.

The above two examples of the k -path testing and satisfiability problems, the rich literature about polynomial testing and many other observations have motivated us to develop a new theory of testing monomials in polynomials represented by arithmetic circuits or even simpler structures. The monomial testing problem is related to, and somehow complements with, the low degree testing and the identity testing of polynomials. We want to investigate various complexity aspects of the monomial testing problem and its variants with two folds of objectives. One is to understand how this problem relates to critical problems in complexity, and if so to what extent. The other is to exploit possibilities of applying algebraic properties of polynomials to the study of those critical problems. As a first step, Chen and Fu [7] have proved a series of results: The multilinear monomial testing problem for $\Pi\Sigma\Pi$ polynomials is NP-hard, even when each clause has at most three terms. The testing problem for $\Pi\Sigma$ polynomials is in P, and so is the testing for two-term $\Pi\Sigma\Pi$ polynomials. However, the testing for a product of one two-term $\Pi\Sigma\Pi$ polynomial and another $\Pi\Sigma$ polynomial is NP-hard. This type of polynomial products is, more or less, related to the polynomial factorization problem. We have also proved that testing c -monomials for two-term $\Pi\Sigma\Pi$ polynomials is NP-hard for any $c > 2$, but the same testing is in P for $\Pi\Sigma$ polynomials. Finally, two parameterized algorithms have been devised for three-term $\Pi\Sigma\Pi$ polynomials and products of two-term

$\Pi\Sigma\Pi$ and $\Pi\Sigma$ polynomials. These results have laid a basis for further study about testing monomials.

1.2 Contributions and Methods

The major contributions of this paper are two pairs of algorithms. For the first pair, we prove that there is a randomized $O^*(p^k)$ time algorithm for testing p -monomials in an n -variate polynomial of degree k represented by an arithmetic circuit, while a deterministic $O^*(6.4^k + p^k)$ time algorithm is devised when the circuit is a formula, here p is a given prime number. The first algorithm extends two recent algorithms for testing multilinear monomials, the $O^*(2^{3k/2})$ algorithm by Koutis [15] and the $O(2^k)$ algorithm by Williams [22]. Koutis [15] initiated the application of group algebra $Z_2[Z_2^k]$ to randomized testing of multilinear monomials in a polynomial. Williams [22] incorporated the randomized Schwartz-Zippel polynomial identity testing with the group algebra $\text{GF}(2^\ell)[Z_p^k]$ for some relatively small ℓ in comparison with k to achieve the design of his algorithm. The success of applying group algebra to designing multilinear monomial testing algorithms is based on two simple but elegant properties found by Koutis, by which annihilating non-multilinear monomials is possible via replacements of variables by vectors in Z_2^k . When extending the group algebra from $Z_2[Z_2^k]$ to $Z_p[Z_p^k]$ for a given prime p these two properties, as addressed in Section 3, are fortunately no longer valid. To make the matter worse, the Schwartz-Zippel algorithm is not applicable to the larger algebra due to the lack of these two properties. Nevertheless, we find new characteristics about $Z_p[Z_p^k]$ and integrate these with a more powerful randomized polynomial identity testing algorithm by Agrawal and Biswas [1] to accomplish the design of our algorithm. Our deterministic algorithm is obtained via derandomizing the two random processes involved in the first algorithm: deterministic selection of a set of linearly independent vectors for an unknown monomial to guarantee its survivability from vector replacements; and deterministic polynomial identity testing. The first part is realized with the perfect hashing functions by Chen *et al.* [8], while the second is carried out by the Raz and Shpilka [19] algorithm for noncommutative polynomials.

For the second pair of our algorithms, we present a deterministic $O^*(2^k)$ time algorithm for testing multilinear monomials in $\Pi_m \Sigma_2 \Pi_t \times \Pi_k \Pi_3$ polynomials, while a randomized $O^*(1.5^k)$ algorithm is given for these polynomials. It has been proved in Chen and Fu [7] that testing multilinear monomials in $\Pi_m \Sigma_2 \Pi_t$ or $\Pi_k \Pi_3$ polynomials is solvable in polynomial time. However, the problem becomes NP-hard for $\Pi_m \Sigma_2 \Pi_t \times \Pi_k \Pi_3$ polynomials. Our two algorithms use the quadratic algorithm by Chen and Fu [7] for testing multilinear monomials in $\Pi_m \Sigma_2 \Pi_t$ polynomials as the base case algorithm. Both new algorithms improve the $O^*(3^k)$ algorithm in [7].

Finally, we prove that testing some special types of multilinear monomials is W[1]-hard, giving evidence that testing for specific monomials is not fixed-parameter tractable. One shall notice that difference between the general monomial testing and the specific monomial testing. The former asks for the existence

of "any one" from a set of possibly many monomials that are needed. The latter asks for "a specific one" from the set.

1.3 Organization

The rest of the paper is organized as follows. In Section 2, we introduce the necessary notations and definitions. In Section 3, we prove new properties about the group algebra $Z_p[Z_p^k]$ to help annihilate any monomials that are not p -monomials. These properties are then integrated with the randomized polynomial identity testing over a finite field to help design the randomize p -monomial testing algorithm. In Section 4, the two randomized processes involved in the randomized algorithm obtained in the previous section will be derandomized for polynomials represented by formulas. The success is based on combining deterministic construction of perfect hashing functions with deterministic noncommunicative polynomial identity testing. Section 5 first presents a deterministic parameterized algorithm for testing multilinear monomials in $\Pi_m \Sigma_2 \Pi_t \times \Pi_k \Sigma_3$ polynomials, and then gives a more efficient randomized parameterized algorithm for these polynomials. Finally, we show in Section 5 that testing some special type of multilinear monomials, called k -clique monomials, is $W[1]$ -hard.

2 Preliminaries

2.1 Notations and Definitions

For variables x_1, \dots, x_n , let $\mathcal{P}[x_1, \dots, x_n]$ denote the communicative ring of all the n -variate polynomials with coefficients from a finite field \mathcal{P} . For $1 \leq i_1 < \dots < i_k \leq n$, $\pi = x_{i_1}^{j_1} \dots x_{i_k}^{j_k}$ is called a monomial. The degree of π , denoted by $\deg(\pi)$, is $\sum_{s=1}^k j_s$. π is multilinear, if $j_1 = \dots = j_k = 1$, i.e., π is linear in all its variables x_{i_1}, \dots, x_{i_k} . For any given integer $c \geq 1$, π is called a c -monomial, if $1 \leq j_1, \dots, j_k < c$.

An arithmetic circuit, or circuit for short, is a direct acyclic graph with $+$ gates of unbounded fan-in, \times gates of fan-in two, and all terminals corresponding to variables. The size, denoted by $s(n)$, of a circuit with n variables is the number of gates in it. A circuit is called a formula, if the fan-out of every gate is at most one, i.e., its underlying direct acyclic graph is a tree.

By definition, any polynomial $F(x_1, \dots, x_n)$ can be expressed as a sum of a list of monomials, called the sum-product expansion. The degree of the polynomial is the largest degree of its monomials in the expansion. With this expression, it is trivial to see whether $F(x_1, \dots, x_n)$ has a multilinear monomial, or a monomial with any given pattern. Unfortunately, this expression is essentially problematic and infeasible to realize, because a polynomial may often have exponentially many monomials in its expansion.

In general, a polynomial $F(x_1, \dots, x_n)$ can be represented by a circuit or some even simpler structure as defined in the following. This type of represen-

tation is simple and compact and may have a substantially smaller size, say, polynomially in n , in comparison with the number of all monomials in the sum-product expansion. The challenge is how to test whether $F(x_1, \dots, x_n)$ has a multilinear monomial, or some other needed monomial, efficiently without unfolding it into its sum-product expansion?

Throughout this paper, the $O^*(\cdot)$ notation is used to suppress $\text{poly}(n, k)$ factors in time complexity bounds.

Definition 1 *Let $F(x_1, \dots, x_n) \in \mathcal{P}[x_1, \dots, x_n]$ be any given polynomial. Let $m, s, t \geq 1$ be integers.*

- *$F(x_1, \dots, x_n)$ is said to be a $\Pi_m \Sigma_s \Pi_t$ polynomial, if $F(x_1, \dots, x_n) = \prod_{i=1}^t F_i$, $F_i = \sum_{j=1}^{r_i} X_{ij}$ and $1 \leq r_i \leq s$, and X_{ij} is a product of variables with $\deg(X_{ij}) \leq t$. We call each F_i a clause. Note that X_{ij} is not a monomial in the sum-product expansion of $p(x_1, \dots, x_n)$ unless $m = 1$. To differentiate this subtlety, we call X_{ij} a term.*
- *In particular, we say $F(x_1, \dots, x_n)$ is a $\Pi_m \Sigma_s$ polynomial, if it is a $\Pi_m \Sigma_s \Pi_1$ polynomial. Here, each clause in f is a linear addition of single variables. In other word, each term has degree 1.*
- *$F(x_1, \dots, x_n)$ is called a $\Pi_m \Sigma_s \Pi_t \times \Pi_k \Sigma_\ell$ polynomial, if $F(x_1, \dots, x_n) = f_1 \cdot f_2$ such that f_1 is a $\Pi_m \Sigma_s \Pi_t$ polynomial and f_2 is a $\Pi_k \Sigma_\ell$ polynomial.*

When no confusion arises from the context, we use $\Pi \Sigma \Pi$ and $\Pi \Sigma$ to stand for $\Pi_m \Sigma_s \Pi_t$ and $\Pi_m \Sigma_s$, respectively.

2.2 The Group Algebra $F[Z_p^k]$

For any prime p and integer $k \geq 2$, we consider the group Z_p^k with the multiplication \cdot defined as follows. For k -dimensional column vectors $\vec{x}, \vec{y} \in Z_p^k$ with $\vec{x} = (x_1, \dots, x_k)^T$ and $\vec{y} = (y_1, \dots, y_k)^T$,

$$\vec{x} \cdot \vec{y} = (x_1 + y_1 \pmod{p}, \dots, x_k + y_k \pmod{p}). \quad (1)$$

$\vec{0} = (0, \dots, 0)^T$ is the zero element in the group. For any field F , the group algebra $F[Z_p^k]$ is defined as follows. Every element $u \in F[Z_p^k]$ is a linear addition of the form

$$u = \sum_{\vec{x} \in Z_p^k, a_{\vec{x}} \in F} a_{\vec{x}} \vec{x}. \quad (2)$$

For any element

$$v = \sum_{\vec{x} \in Z_p^k, b_{\vec{x}} \in F} b_{\vec{x}} \vec{x},$$

We define

$$u + v = \sum_{a_{\vec{x}}, b_{\vec{x}} \in F, \vec{x} \in Z_p^k} (a_{\vec{x}} + b_{\vec{x}} \pmod{p}) \vec{x}, \text{ and} \quad (3)$$

$$u \cdot v = \sum_{a_{\vec{x}}, b_{\vec{y}} \in F, \text{ and } \vec{x}, \vec{y} \in Z_p^k} (a_{\vec{x}} b_{\vec{y}} \pmod{p}) (\vec{x} \cdot \vec{y}). \quad (4)$$

For any scalar $w \in F$,

$$wu = a \left(\sum_{\vec{x} \in Z_p^k, a_{\vec{x}} \in F} a_{\vec{x}} \vec{x} \right) = \sum_{\vec{x} \in Z_p^k, a_{\vec{x}} \in F} (wa_{\vec{x}} \pmod{p}) \vec{x}. \quad (5)$$

The zero element in $F[Z_p^k]$ is the one as represented in expression (2) with zero coefficients in F :

$$\mathbf{0} = \sum_{\vec{x} \in Z_p^k} 0\vec{x} = \mathbf{0}\vec{0}. \quad (6)$$

The identity element in $F[Z_p^k]$ is

$$\mathbf{1} = \mathbf{1}\vec{0} = \vec{0}. \quad (7)$$

For any vector $\vec{v} = (v_1, \dots, v_k)^T \in Z_p^k$, for $i \geq 0$, let

$$(\vec{v})^i = (iv_1 \pmod{p}, \dots, iv_k \pmod{p})^T.$$

In particular, we have

$$(\vec{v})^0 = (\vec{v})^p = \vec{0}.$$

When it is clear from the context, we will simply use xy and $x + y$ to stand for $xy \pmod{p}$ and $x + y \pmod{p}$, respectively.

3 Randomized Testing of p -Monomials

Group algebra $Z_2[Z_2^k]$ was first used by Koutis [15] and later by Williams [22] to devise a randomized $O^*(2^k)$ time algorithm to test multilinear monomials in n -variate polynomials represented by arithmetic circuits. We shall extend $Z_2[Z_2^k]$ to $Z_p[Z_p^d]$ to test p -monomials for some $d > k$. Two key properties in $Z_2[Z_2^k]$, as first found by Koutis [15], that are crucial to multilinear monomial testing are unfortunately no longer valid in $Z_p[Z_p^d]$. Instead, we establish new properties in Lemmas 4 and 5. Also, the Schwartz-Zippel algorithm [17] for randomized polynomial identity testing adopted by Williams [22] is not applicable to our case. Instead, we have to use a more advanced randomized polynomial identity testing algorithm, the Agrawal and Biswas algorithm [1].

Let p be a prime number. Following conventional notations in linear algebra, for any vectors $\vec{v}_1, \dots, \vec{v}_t \in Z_p^k$ with $k \geq 1$ and $t \geq 1$, let $\text{span}(\vec{v}_1, \dots, \vec{v}_t)$ be the linear space spanned by these vectors. That is,

$$\text{span}(\vec{v}_1, \dots, \vec{v}_t) = \{a_1\vec{v}_1 + \dots + a_t\vec{v}_t \mid a_1, \dots, a_t \in Z_p\}.$$

We first give two simple properties about $(\text{mod } p)$ operation.

Lemma 2 *For any $x, y \in Z_p$, we have $(x + y)^p = x^p + y^p \pmod{p}$.*

Proof $(x + y)^p = \sum_{i=0}^p \binom{p}{i} x^{p-i} y^i = x^p + y^p + \sum_{i=1}^{p-1} \binom{p}{i} x^{p-i} y^i$. Since p is prime, $\binom{p}{i}$ has a factor p , implying $\binom{p}{i} = 0 \pmod{p}$, $1 \leq i \leq p-1$. Hence, $(x + y)^p = x^p + y^p \pmod{p}$. \square

Lemma 3 *For any $x, y \in Z_p$, we have $((p-1)x + y)^p = (p-1)x^p + y^p \pmod{p}$.*

Proof By Lemma 2, $((p-1)x + y)^p \equiv (p-1)^p x^p + y^p \pmod{p}$. By Fermat's Little Theorem, $(p-1)^p = (p-1) \pmod{p}$. Thus, $((p-1)x + y)^p = (p-1)x^p + y^p \pmod{p}$. \square

The first crucial, though simple, property observed by Koustis [15] about testing multilinear monomials is that replacing any variable x by $(\vec{v} + \vec{0})$ will annihilate x^t for any $t \geq 2$, where $\vec{v} \in Z_2^k$ and \vec{v}_0 is the zero vector. This property is not valid in $Z_p[Z_p^d]$. However, we shall prove the following lemma that helps annihilate any monomials that are not p -monomials.

Lemma 4 *Let $\vec{v}_0 \in Z_p^d$ be the zero vector and $\vec{v}_i \in Z_p^d$ be any vector. Then, we have*

$$((p-1)\vec{v}_i + \vec{v}_0)^p = \vec{0}, \quad (8)$$

i.e., the zero element in $Z_p[Z_p^d]$.

Proof By Lemma 3, we have $((p-1)\vec{v}_i + \vec{v}_0)^p = (p-1)(\vec{v}_i)^p + (\vec{v}_0)^p \pmod{p} = (p-1)\vec{v}_0 + \vec{v}_0 = p\vec{v}_0 \pmod{p} = \vec{0}$. \square

The second crucial property found by Koutis [15] has two parts: (a) Replacing variables x_{i_j} in a multilinear monomial $x_{i_1} \dots x_{i_k}$ with $(\vec{v}_{i_j} + \vec{v}_0)$ will annihilate the monomial, if the vectors \vec{v}_{i_j} are linearly dependent in Z_2^k . (b) If these vectors are linearly independent, then the sum-product expansion of the monomial after the replacements will yield a sum of all 2^k vectors in Z_2^k . However, neither (a) nor (b) is in general true in $Z_p[Z_p^k]$. Fortunately, we have the following lemma, though not as "structurally" perfect as (b).

Lemma 5 Let $x_1^{m_1} \cdots x_t^{m_t}$ be any given p -monomial of degree k . If vectors $\vec{v}_1, \dots, \vec{v}_t \in Z_p^d$ are linearly independent, then there are nonzero coefficients $c_i \in Z_p$ and distinct vector $\vec{u}_j \in Z_p^d$ such that

$$((p-1)\vec{v}_1 + \vec{v}_0)^{m_1} \cdots ((p-1)\vec{v}_t + \vec{v}_0)^{m_t} = c_1 \vec{0} + \left(\sum_{i=2}^{(m_1+1)(m_2+1)\cdots(m_t+1)} c_i \vec{u}_i \right)$$

where $c_1 = 1$.

Proof

$$\begin{aligned} & ((p-1)\vec{v}_1 + \vec{v}_0)^{m_1} \cdots ((p-1)\vec{v}_t + \vec{v}_0)^{m_t} \\ &= \left(\sum_{i_1=0}^{m_1} \binom{m_1}{i_1} (p-1)^{i_1} (\vec{v}_1)^{i_1} \right) \left(\sum_{i_2=0}^{m_2} \binom{m_2}{i_2} (p-1)^{i_2} (\vec{v}_2)^{i_2} \right) \cdots \left(\sum_{i_t=0}^{m_t} \binom{m_t}{i_t} (p-1)^{i_t} (\vec{v}_t)^{i_t} \right) \\ &= \sum_{i_1=0}^{m_1} \sum_{i_2=0}^{m_2} \cdots \sum_{i_t=0}^{m_t} \binom{m_1}{i_1} \binom{m_2}{i_2} \cdots \binom{m_t}{i_t} (p-1)^{i_1+i_2+\cdots+i_t} (\vec{v}_1)^{i_1} (\vec{v}_2)^{i_2} \cdots (\vec{v}_t)^{i_t} \end{aligned} \quad (10)$$

As noted in the previous section, in the vector space Z_p^d , we have

$$(\vec{v}_1)^{i_1} (\vec{v}_2)^{i_2} \cdots (\vec{v}_t)^{i_t} = i_1 \vec{v}_1 + i_2 \vec{v}_2 + \cdots + i_t \vec{v}_t. \quad (11)$$

Since $\vec{v}_1, \vec{v}_2, \dots, \vec{v}_t$ are linearly independent, by expression (11) we have

$$(\vec{v}_1)^{i_1} (\vec{v}_2)^{i_2} \cdots (\vec{v}_t)^{i_t} = \vec{0} \quad \text{iff} \quad i_1 = i_2 = \cdots = i_t = 0. \quad (12)$$

The linear independence of $\vec{v}_1, \vec{v}_2, \dots, \vec{v}_t$ implies that any non-empty subset of these vectors are also linearly independent. Similar to expression (12), this further implies that, for any $0 \leq j_i \leq m_i$, $i = 1, 2, \dots, t$,

$$(\vec{v}_1)^{i_1} (\vec{v}_2)^{i_2} \cdots (\vec{v}_t)^{i_t} = (\vec{v}_1)^{j_1} (\vec{v}_2)^{j_2} \cdots (\vec{v}_t)^{j_t} \quad \text{iff} \quad i_1 = j_1, i_2 = j_2, \dots, \text{ and } i_t = j_t$$

Furthermore, since p is prime and $m_i \in Z_p$, we have

$$\begin{aligned} c(i_1, i_2, \dots, i_t) &= \binom{m_1}{i_1} \binom{m_2}{i_2} \cdots \binom{m_t}{i_t} (p-1)^{i_1+i_2+\cdots+i_t} \pmod{p} \\ &\neq 0 \pmod{p} \end{aligned} \quad (14)$$

Combining expressions (13) and (14), we have

$$\begin{aligned} & ((p-1)\vec{v}_1 + \vec{v}_0)^{m_1} \cdots ((p-1)\vec{v}_t + \vec{v}_0)^{m_t} \\ &= 1\vec{0} + \left(\sum_{0 \leq i_j \leq m_j, 0 \leq j \leq t, \text{ and } i_1+i_2+\cdots+i_t > 0} c(i_1, i_2, \dots, i_t) \cdot ((\vec{v}_1)^{i_1} (\vec{v}_2)^{i_2} \cdots (\vec{v}_t)^{i_t}) \right) \end{aligned}$$

In the above expression (15), all the coefficients are nonzero, and all the $(m_1 + 1)(m_2 + 1) \cdots (m_t + 1) \leq p^k$ vectors are distinct. Hence, expression (9) is obtained. \square

Remark. Lemma 5 guarantees that replacing variables in a p -monomial by linearly independent vectors will prevent the monomial from being annihilated. Note that the total number of distinct vectors in expression 9 is at most p^k .

Lemmas 4 and 5 have laid a basis for designing randomized algorithms to test p -monomials. One additional help will be drawn from randomized polynomial identity testing over a finite field. We are ready to present the algorithm and show how to integrate group algebra with polynomial identity testing to aid our design. To simplify description, we assume, like in Koutis [15] and Williams [22], that the degree of p -monomials in a polynomial is at least k , provided that such monomials exist. Otherwise, we can simply multiply some new variables to the given polynomial to satisfy the requirement.

Theorem 6 *Let p be a prime number. Let $F(x_1, x_2, \dots, x_n)$ be an n -variate polynomial of degree k represented by an arithmetic circuit C of size $s(n)$. There is a randomized $O^*(p^k)$ time algorithm to test with high probability whether F has a p -monomial of degree k in its sum-product expansion.*

Proof Let $d = k + \log_p k + 1$, we consider the group algebra $Z_p[Z_p^d]$. As in Williams [22], we first expand the circuit C to a new circuit C' as follows. For each multiplication gate g_i , we attach a new gate g'_i that multiplies the output of g_i with a new variable y_i , and feed the output of g'_i to the gate that reads the output of g_i . Assume that C has h multiplications gates. Then, C' will have h new multiplications gates corresponding to new variables y_1, y_2, \dots, y_h . Let $F'(y_1, y_1, \dots, y_h, x_1, x_2, \dots, x_n)$ be the new polynomial represented by C' . The algorithm for testing whether F has a p -monomial of degree k is given in the following.

Algorithm RT-MLM (Randomized Testing of Multilinear Monomials):

1. Select uniform random vectors $\vec{v}_1, \dots, \vec{v}_n \in Z_p^d - \{\vec{0}\}$.
2. Replace each variable x_i with $(\vec{v}_i + \vec{v}_0)$, $1 \leq i \leq n$.
3. Use C' to calculate

$$F'(y_1, \dots, y_h, (\vec{v}_1 + \vec{v}_0), \dots, (\vec{v}_n + \vec{v}_0)) = \sum_{j=1}^{2^d} f_j(y_1, \dots, y_h)(\vec{z}_j)$$

where each f_j is a polynomial of degree k over the finite field Z_p , and \vec{z}_j with $1 \leq j \leq 2^d$ are the 2^d distinct vectors in Z_p^d .

4. Perform polynomial identity testing with the Agrawal and Biswas algorithm [1] for every f_j over Z_p . Return "yes" if one of them is not identical to zero, or "no" otherwise.

It follows from Lemma 4 that all monomials that are not p -monomials in F (and hence in F') will become zero, when variables x_i is replaced by $(\vec{v}_i + \vec{v}_0)$ at Step ii. We shall estimate that with high probability some p -monomials will

survive from those replacements, i.e., will not become the zero element $\mathbf{0}$ in $Z_p[Z_p^d]$.

Consider any given p -monomial $\pi = x_{i_1}^{m_1} \cdots x_{i_t}^{m_t}$ of degree k with $1 \leq m_i < p$ and $k = m_1 + \cdots + m_t$, $i = 1, \dots, t$. For any $1 \leq j \leq t$,

$$\Pr [\vec{v}_j \in \text{span}(\vec{v}_{i_1}, \dots, \vec{v}_{i_{j-1}})] = \frac{p^{j-1}}{p^d},$$

since $|\text{span}(\vec{v}_{i_1}, \dots, \vec{v}_{i_{j-1}})| = p^{j-1}$ and $|Z_p^d| = p^d$. Hence,

$$\begin{aligned} & \Pr [(\exists j \in \{1, \dots, t\})[\vec{v}_{i_j} \in \text{span}(\vec{v}_{i_1}, \dots, \vec{v}_{i_{j-1}})]] \\ &= \Pr \left[[\vec{v}_1 = \vec{\mathbf{0}}] \vee [\vec{v}_{i_2} \in \text{span}(\vec{v}_{i_1})] \vee \cdots \vee [\vec{v}_{i_t} \in \text{span}(\vec{v}_{i_1}, \dots, \vec{v}_{i_{t-1}})] \right] \\ &\leq \Pr[\vec{v}_1 = \vec{\mathbf{0}}] + \Pr[\vec{v}_{i_2} \in \text{span}(\vec{v}_{i_1})] + \cdots + \Pr[\vec{v}_{i_t} \in \text{span}(\vec{v}_{i_1}, \dots, \vec{v}_{i_{t-1}})] \\ &= \frac{p^0}{p^d} + \frac{p^1}{p^d} + \cdots + \frac{p^{t-1}}{p^d} \leq t \frac{p^{t-1}}{p^d} \\ &\leq k \frac{p^{k-1}}{p^{k+\log_p k+1}} \leq \frac{1}{p^2} \leq \frac{1}{4}. \end{aligned} \tag{17}$$

Because $\vec{v}_{i_1}, \dots, \vec{v}_{i_t}$ are linearly independent iff there is no $\vec{v}_{i_j} \in \text{span}(\vec{v}_{i_1}, \dots, \vec{v}_{i_{j-1}})$, by expression (17) the probability that $\vec{v}_{i_1}, \dots, \vec{v}_{i_t}$ are linearly independent is at least $\frac{3}{4}$. This implies, by Lemma 5, that the monomial π will survive from the replacements at Step ii with probability at least $\frac{3}{4}$. Furthermore, by expression (9) in Lemma 5,

$$((p-1)\vec{v}_1 + \vec{v}_0)^{m_1} \cdots ((p-1)\vec{v}_t + \vec{v}_0)^{m_t} = \sum_{i=1}^{p^k} c(\pi)_i \vec{u}_i(\pi), \tag{18}$$

where $c(\pi)_i$ are coefficients in Z_p such that $(m_1+1)(m_2+1) \cdots (m_t+1)$ of them are nonzero, and $\vec{u}_i(\pi)$ are distinct vectors in Z_p^d . Let $\psi(\pi)$ be the product of the new variables y_j that are added with respect to the gates in C such that those gates produce the monomial π . Then, $\psi(\pi)$ is a monomial that is generated by C' . Hence, at Step iii, by expression (18) F' will have monomials respect to π as given in the following expansion:

$$\begin{aligned} \phi(\pi) &= \psi(\pi) \cdot ((p-1)\vec{v}_1 + \vec{v}_0)^{m_1} \cdots ((p-1)\vec{v}_t + \vec{v}_0)^{m_t} \\ &= \sum_{i=1}^{p^k} c(\pi)_i \cdot \psi(\pi) \cdot \vec{u}_i(\pi). \end{aligned} \tag{19}$$

Let \mathcal{S} be the set of all those p -monomials that survive from the variable replacements. Then,

$$F'(y_1, \dots, y_h, (\vec{v}_1 + \vec{v}_0), \dots, (\vec{v}_n + \vec{v}_0)) = \sum_{\pi \in \mathcal{S}} \phi(\pi)$$

$$\begin{aligned}
&= \sum_{\pi \in \mathcal{S}} \left(\sum_{i=1}^{p^k} c(\pi)_i \cdot \psi(\pi) \cdot \vec{u}_i(\pi) \right) \\
&= \sum_{j=1}^{2^d} \left(\sum_{\pi \in \mathcal{S} \text{ and } \vec{z}_j = \vec{u}_i(\pi)} c(\pi)_i \cdot \psi(\pi) \right) \cdot \vec{z}_j \tag{20}
\end{aligned}$$

Let

$$f_j(y_1, \dots, y_h) = \sum_{\pi \in \mathcal{S} \text{ and } \vec{z}_j = \vec{u}_i(\pi)} c(\pi)_i \cdot \psi(\pi),$$

then the degree k polynomial with respect to \vec{z}_j is obtained for F' in expression (16).

Recall that when constructing the circuit C' , each new gate is associated with a new variable. This means that for any two monomials π' and π'' in F , we have $\psi(\pi') \neq \psi(\pi'')$. This implies that we cannot add $c(\pi') \cdot \psi(\pi')$ to $c(\pi'') \cdot \psi(\pi'')$ in f_j . Thus, the possibility of a "zero-sum" of coefficients from different surviving monomials is completely avoided during the construction of f_j . Therefore, conditioned on that \mathcal{S} is not empty, F' must not be identical to zero, i.e., there exists at least one f_j that is not identical to zero. At Step iv, we use the randomized algorithm by Agrawal and Biswas [1] to test whether f_j is identical to zero. It follows from Theorem 4.6 in Agrawal and Biswas [1] that this testing can be done with probability at least $\frac{5}{6}$ in time polynomially in $s(n)$ and $\log q$. Since \mathcal{S} is not empty with probability at least $\frac{3}{4}$, the probability of overall success of testing whether F has a p -monomial is at least $\frac{5}{8}$.

Finally, we address the issues about how to calculate F' and the time needed to do so. Naturally, every element in the group algebra $Z_p[Z_p^d]$ can be represented by a vector in $Z_p^{p^d}$. Adding two elements in $Z_p[Z_p^d]$ is equivalent to adding the two corresponding vectors in $Z_p^{p^d}$, and the latter can be done in $O(p^d \log p)$ time via component-wise sum. In addition, multiplying two elements in $Z_p[Z_p^d]$ is equivalent to multiplying the two corresponding vectors in $Z_p^{p^d}$, and the latter can be done in $O(dp^d \log^2 p)$ with the help of a similar Fast Fourier Transform style algorithm as in Williams [22]. Calculating F' consists of $s(n)$ arithmetic operations of either adding or multiplying two elements in $Z_p[Z_p^d]$ based on the circuit C or C' . Hence, the total time needed is $O(s(n)dp^d \log^2 p)$. At Step iv, we run the Agrawal and Biswas [1] algorithm to F' to simultaneously testing whether there is one f_j such that f_j is not identical to zero. We choose a probability $\frac{5}{6}$, the by Theorem 4.6 in Agrawal and Biswas [1], this testing can be done in $O^*((s(n))^4 n^4 \log^2 p)$ time, suppressing a $\text{poly}(\log s(n), \log n, \log \log p)$ factor. Recall that $d = k + \log_p k + 1$. The total time for the entire algorithm is $O^*(p^k)$. \square

4 Derandomization

In this section, we turn our attention to formulas instead of general arithmetic circuits and shall design a deterministic algorithm to test p -monomials for polynomials represented by a formula. Recall that the algorithm RT-MLM has only two randomized processes at Step i to select n uniform random variables and at Step iv to test whether one f_j from F' is identical to zero over Z_p . In this section, we shall derandomize these two randomized processes respectively with the help of two advanced techniques of perfect hashing by Chen *et al.* [8] and Naor *et al.* [18] and noncommunicative multivariate polynomial identity testing by Raz and Shpilka [19].

Let n and k be two integers such that $1 \leq k \leq n$. Let $\mathcal{A} = \{1, 2, \dots, n\}$ and $\mathcal{K} = \{1, 2, \dots, k\}$. A k -coloring of the set \mathcal{A} is a function from \mathcal{A} to \mathcal{K} . A collection \mathcal{F} of k -colorings of \mathcal{A} is a (n, k) -family of *perfect hashing functions* if for any subset W of k elements in \mathcal{A} , there is a k -coloring $h \in \mathcal{F}$ that is injective from W to \mathcal{K} , i.e., for any $x, y \in W$, $h(x)$ and $h(y)$ are distinct elements in \mathcal{K} .

Theorem 7 *Let p be a prime number. Let $F(x_1, x_2, \dots, x_n)$ be an n -variate polynomial of degree k represented by a formula C of size $s(n)$. There is a deterministic $O(6.4^k + p^k)$ time algorithm to test whether F has a p -monomial of degree k in its sum-product expansion.*

Proof As in the proof of Theorem 6, we consider the group algebra $Z_p[Z_p^k]$. Here, we do not need to expand the dimension k to $d > k$. We also construct a new formula C' from C by adding new variable y_i for each multiplication gate g_i in the same way as what we did for Theorem 6. Assume that C has h many multiplication gates, then C' will have h new multiplication gates corresponding to new variables y_1, y_2, \dots, y_h . The algorithm for testing whether F has a p -monomial of degree k is given as follows.

Algorithm DT-MLM (Deterministic Testing of Multilinear Monomials):

1. Construct with the algorithm by Chen *et al.* [8] an (n, k) -family of perfect hashing functions \mathcal{H} of size $O(6.4^k \log^2 n)$.
2. Select k linearly independent vectors $\vec{v}_1, \dots, \vec{v}_k \in Z_p^k$. (No randomization is needed at this step.)
3. For each perfect hashing function $\tau \in \mathcal{H}$ do
 - a. For each variable x_i , replace it by $(\vec{v}_{\tau(i)} + \vec{v}_0)$.
 - b. Use C' to calculate

$$\begin{aligned}
& F'(y_1, \dots, y_h, (\vec{v}_1 + \vec{v}_0), \dots, (\vec{v}_n + \vec{v}_0)) \\
&= \sum_{j=1}^{2^k} f_j(y_1, y_2, \dots, y_h) \cdot \vec{z}_j, \tag{21}
\end{aligned}$$

- where each f_j is a polynomial of degree k over the finite field Z_p , and vectors \vec{z}_j with $1 \leq j \leq 2^k$ are the 2^k distinct vectors in Z_p^k .
- c. Perform polynomial identity testing with the Raz and Shpilka algorithm [19] for every f_j over Z_p . Stop and return "yes" if one of them is not identical to zero.
 - iv. If all perfect hashing functions in \mathcal{H} have been tried without returning "yes", then stop and output "no".

By Chen *et al.* [8], Step i can be done in $O(6.4^k n \log^2 n)$ times. Step ii can be easily done in $O(k^2 \log p)$ time.

It follows from Lemma 4 that all those monomials that are not p -monomials in F , and hence in F' , will be annihilated, when variables x_i are replaced by $(\vec{v}_i + \vec{v}_0)$ at Step iii.a.

Consider any given p -monomial $\pi = x_{i_1}^{m_1} \cdots x_{i_t}^{m_t}$ of degree k with $1 \leq m_i < p$ and $k = m_1 + \cdots + m_t$, $i = 1, \dots, t$. Because of the nature of \mathcal{H} , there is at least one perfect hashing function τ in \mathcal{H} such that $\tau(i_{j'}) \neq \tau(i_{j''})$ if $i_{j'} \neq i_{j''}$, $1 \leq j', j'' \leq t \leq k$. This means that $\vec{v}_{\tau(i_1)}, \dots, \vec{v}_{\tau(i_t)}$ are distinct and hence linearly independent. By Lemma 5, π will survive from the replacements at Step iii.a. Let \mathcal{S} be the set of all surviving p -monomials. Following the same analysis as in the proof of Theorem 6, we have F' that is not identical to zero if \mathcal{S} is not empty. That is, there is at least one f_j that is not identical to zero, if \mathcal{S} is not empty. Moreover, the time needed for calculating F' is $O(kp^k \log^2 p)$.

We now consider imposing noncommunicativity on C' as follows. Inputs to an arithmetic gate are ordered so that the formal expressions $y_{i_1} \cdot y_{i_2} \cdots y_{i_r}$ and $y_{j_1} \cdot y_{j_2} \cdots y_{j_l}$ are the same iff $r = l$ and $i_q = j_q$ for $q = 1, \dots, r$. Finally, we use the algorithm by Raz and Shpilka [19] to test whether $f_j(y_1, \dots, y_h)$ is identical to zero or not. This can be done in time polynomially in $s(n)$ and n , since f_j is a non-communicative polynomial represented by a formula.

Combining the above analysis, the total time of the algorithm DT-MLM is $O(6.4^k n \log^2 n + kp^k(s(n)n)^{O(1)} \log^2 p) = O^*(6.4^k + p^k)$. \square

5 $\Pi_m \Sigma_2 \Pi_t \times \Pi_k \Sigma_3$ Polynomials

It has been proved by Chen and Fu [7] that the problem of testing monomials in $\Pi_m \Sigma_s$ polynomials is solvable in $(ms\sqrt{m+s})$ time, and in $\Pi_m \Sigma_2 \Pi_t$ polynomials is in $O((mt)^2)$ time. On the other hand, it has also been proved by in [7] that the problem for $\Pi_m \Sigma_3$ and $\Pi_m \Sigma_2 \Pi_t \times \Pi_k \Sigma_3$ polynomials is respectively NP-complete. Moreover, a $O(tm^2 1.7751^m)$ time algorithm was obtained for $\Pi_m \Sigma_3 \Pi_t$ polynomials, and so was a $O((mt)^2 3^k)$ algorithm obtained for $\Pi_m \Sigma_2 \Pi_t \times \Pi_k \Sigma_3$ polynomials. In this section, we shall devise two parameterized algorithms, one deterministic and the other randomized, for testing multilinear monomials in $\Pi_m \Sigma_2 \Pi_t \times \Pi_k \Sigma_3$ polynomials, improving the $O((mt)^2 3^k)$ upper bound in [7].

Theorem 8 *There is a deterministic algorithm of time $O(((mt + k)^2 + k)2^k)$ to test whether any $\Pi_m \Sigma_2 \Pi_t \times \Pi_k \Sigma_3$ polynomial has a multilinear monomial in its sum-product expansion.*

Proof Let $F = F_1 \cdot F_2$ such that $F_1 = f_1 \cdots f_m$ is a $\Pi_m \Sigma_2 \Pi_t$ polynomial and $F_2 = g_1 \cdots g_k$ is a $\Pi_k \Sigma_3$ polynomial, where $f_i = (T_{i1} + T_{i2})$ and $g_j = (x_{j1} + x_{j2} + x_{j3})$, $1 \leq i \leq m$, $1 \leq j \leq k$.

Consider variable x_{11} in the clause g_1 . We devise a branch and bound process to divide the testing for F into the testing for two new polynomials. We eliminate all x_{11} in g_j for $j = 1, \dots, k$. Let g'_j be the clause resulted from g_j after the eliminating process. Let $h_1 = F_1 \cdot g'_1$, $h_2 = F_1 \cdot x_{11}$, $q = g'_2 \cdots g'_k$. Note that exactly one of the three variable x_{11}, x_{12} and x_{13} in the clause g_1 must be selected to form a monomial (hence a multilinear monomial) for F in the sum-product expansion of F . We have two cases concerning the selection of x_{11} :

(1) x_{11} can not be selected to help form any multilinear monomial. In this case, F has a multilinear monomial, iff $h_1 \cdot q$ has a multilinear monomial.

(2) x_{11} can be selected to form a multilinear monomial. Thus, F has a multilinear monomial, iff $h_2 \cdot q$ has a multilinear monomial.

In either case, the new polynomial is a product of two polynomials with the first being a $\Pi_{m+1} \Sigma_2 \Pi_t$ polynomial and the second a $\Pi_k \Sigma_3$ polynomial. Furthermore, the second is the common q , which has one fewer clause than F_2 .

Let $T(k)$ denote the time for testing multilinear monomials in F . Notice that the eliminating process for x_{11} takes $O(k)$ time. Then, $T(k)$ is bounded as follows

$$T(k) \leq 2T(k-1) + O(k) \leq 2^k(T(0) + O(k)).$$

$T(0)$ is the time to test multilinear monomials in a $\Pi_{m+k} \Sigma_2 \Pi_t$ polynomial with a size of $O(mt + k)$. By the algorithm in [7] for this type of polynomials, $T(0) = O((mt + k)^2)$. Therefore, $T(k) = O(((mt + k)^2 + k)2^k)$. \square

We now show that the upper bound in the above theorem can be further improved via randomization.

Theorem 9 *There is a $O((mt + k)^{2.15^k})$ time randomized algorithm that finds a multilinear monomial for any $\Pi_m \Sigma_2 \Pi_t \times \Pi_k \Sigma_3$ polynomial with probability at least $1 - \frac{1}{e}$ if such monomials exist, or returns "no" otherwise.*

Proof Like in Theorem 8, let $F = F_1 \cdot F_2$ such that $F_1 = f_1 \cdots f_m$ is a $\Pi_m \Sigma_2 \Pi_t$ polynomial and $F_2 = g_1 \cdots g_k$ is a $\Pi_k \Sigma_3$ polynomial with $f_i = (T_{i1} + T_{i2})$ and $g_j = (x_{j1} + x_{j2} + x_{j3})$.

Assume that F has a multilinear monomial π . Then, one of the three variables in g_j must be included in π , $1 \leq j \leq k$. We uniformly select two distinct variables y_{j1} and y_{j2} from g_j , then $g'_j = (y_{j1} + y_{j2})$ contains a desired variable for π with a probability at least $2/3$. Let

$$F' = F_1 \cdot (g'_1 \cdots g'_k),$$

then F' has a multilinear monomial with a probability at least $(\frac{2}{3})^k$. On the other hand, if F does not have any multilinear monomials in its sum-product expansion, then F' must not have any multilinear monomials. Notice that F' is a $\Pi_{m+k}\Sigma_2\Pi_t$ polynomial with a size of $O(mt+k)$. By the algorithm for this type of polynomials by Chen and Fu in [7], one can find a multilinear monomial in F' in time $O((mt+k)^2)$. In other words, the above randomized process will fail to find a multilinear monomial in F with a probability of at most $1 - (\frac{2}{3})^k$ if such monomials exist, or return "no" otherwise.

Repeat the above randomized process $(\frac{3}{2})^k$ many times. If F has multilinear monomials, then these processes will fail to find one with a probability of at most

$$\left[1 - \left(\frac{2}{3}\right)^k\right]^{(\frac{3}{2})^k} < \frac{1}{e}.$$

Hence, the processes will find a multilinear monomial in F with a probability of at least $1 - \frac{1}{e}$. If F does not have any multilinear monomial, then none of these repeated processes will find one in F . The total time of all the repeated processes is $O((mt+k)^2 1.5^k)$. □

It is justified in [7] that the resemblance of $\Pi\Sigma\Pi$ and $\Pi\Sigma$ polynomials with SAT formulas is "superficial". For example, The multilinear monomial testing problem for $\Pi_m\Sigma_3\Pi_1$ polynomials is in P, but 3SAT is NP-complete. As another example to show such superficial resemblance, one might consider to apply Schönning's algorithm for 3SAT [20] to the multilinear monomial testing problem. However, this is problematic. For the 3SAT problem, it is easy to find an unsatisfied 3-clause. On the other hand, for the multilinear monomial testing problem, we do not know which term in which clause leads to a confliction. Therefore, it is difficult to decide the change of the hamming distance between the current solution and any target solution. This difficulty constitutes a major barrier towards applying the Schönning's algorithm to monomial testing.

6 W[1]-Hardness

Although deterministic and randomized parameterized algorithms have been devised for testing monomials in previous three sections as well as in [15, 22, 7], yet we shall prove in this section that testing some special type of monomials in polynomials represented by arithmetic circuits is not fixed-parameter tractable, unless some unlikely collapse occurs in the fixed parameter complexity theory.

One shall notice that difference between the general monomial testing and the specific monomial testing. The former asks for the existence of "any one" from a set of possibly many monomials that are needed. The latter asks for "a specific one" from the set. For example, there may be $2^n - 1$ multilinear monomials in the sum-product expansion of a n -variate polynomials. Testing for any one from these many monomials is certainly different from testing for a specific one, say, $x_1x_3x_7x_{11}$.

Downey and Fellows [10] have established a hierarchy of parameterized complexity, named the W hierarchy, and proved that the k -Clique problem is W[1]-hard.

Definition 10 Let $C = \{i_1, i_2, \dots, i_k\}$ be a set of k positive integers. A k -clique monomial with respect to C is the multilinear monomial $\prod_{1 \leq j < \ell \leq k} x_{i_j i_\ell}$ of degree $\frac{k(k-1)}{2}$.

Theorem 11 It is W[1]-hard to test whether any given n -variate polynomial of degree $\frac{k(k-1)}{2}$ represented by an arithmetic circuit has a k -clique monomial in its sum-product expansion.

Proof We shall reduce the k -clique problem to the k -clique monomial testing problem. Let $G = (V, E)$ be an undirected graph and k an integer parameter. $V = \{v_1, v_2, \dots, v_m\}$ is the set of vertices. Each $(i, j) \in E$ represents the edge connecting vertices v_i and v_j . For each edge $(i, j) \in E$, we define a variable x_{ij} . Let $n = |E|$. We construct a polynomial f with n variables.

$$\begin{aligned} f(G, 1) &= 1, \\ f(G, 2) &= \sum_{(i,j) \in E} x_{ij}, \\ f(G, t+1) &= \sum_{i=1}^m \left(\sum_{(i,j) \in E} x_{ij} \right)^t \cdot f(G, t) \end{aligned}$$

As followed from the above definition, $f(G, k)$ has $n = |E|$ variables and its degree is $\frac{k(k-1)}{2}$. It is easy to see that $f(G, k)$ can be computed by an arithmetic circuit.

If G has a k -clique $A = \{i_1, i_2, \dots, i_k\}$, then there are $\frac{k(k-1)}{2}$ edges connecting any two vertices in A . By definition, $f(G, k)$ has a term $(x_{i_1 i_2} + \dots + x_{i_1 i_k} + \dots + x_{i_{k-1} i_k})^{k-1} \cdot f(G, k-1)$. So, we can select $\pi_1 = x_{i_1 i_2} \dots x_{i_1 i_k}$ from the first factor of this term. By simple induction, we can select a $(k-1)$ -clique monomial of degree $\frac{(k-1)(k-2)}{2}$ with respect to $A - \{i_1\}$. Then, $\pi_1 \cdot \pi_2$ is a k -clique monomial with respect to A . On the other hand, if $f(G, k)$ has a k -clique monomial with respect to A , then by definition, A is a k -clique for G . \square

Acknowledgments

We thank Ioannis Koutis for helping us understand his group algebra in [15]. Bin Fu's research is supported by an NSF CAREER Award, 2009 April 1 to 2014 March 31.

References

- [1] Manindra Agrawal and Somenath Biswas, Primality and identity testing via Chinese remaindering, *Journal of the ACM* 50(4): 429-443, 2003.
- [2] Manindra Agrawal, Neeraj Kayal and Nitin Saxena, PRIMES is in P, *Ann. of Math*, 160(2): 781-793, 2004.
- [3] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy, Proof verification and the hardness of approximation problems, *Journal of the ACM* 45 (3): 501-555, 1998.
- [4] Bengt Aspvall, Michael F. Plass and Robert E. Tarjan, A linear-time algorithm for testing the truth of certain quantified boolean formulas, *Information Processing Letters* 8 (3): 121-123, 1979.
- [5] Richard Beigel, The polynomial method in circuit complexity, *Proceedings of the Eighth Conference on Structure in Complexity Theory*, pp. 82-95, 1993.
- [6] Nader H. Bshouty, Zhixiang Chen, Scott E. Decatur, and Steve Homer, One the learnability of Z_N -DNF formulas, *Proceedings of the Eighth Annual Conference on Computational Learning Theory (COLT 1995)*, Santa Cruz, California, USA. ACM, 1995, pp. 198-205.
- [7] Zhixiang Chen and Bin Fu, The complexity of testing monomials in multivariate polynomials, submitted for publication, June 2010.
- [8] Jianer Chen, Songjian Lu, Sing-Hoi Sze and Fenghui Zhang, Improved algorithms for path, matching, and packing problems, *SODA*, pp. 298-307, 2007.
- [9] Zhi-Zhong Chen and Ming-Yang Kao, Reducing randomness via irrational numbers, *SIAM J. Comput.* 29(4): 1247-1256, 2000.
- [10] R.G. Downey and M.R. Fellows, Fixed parameter tractability and completeness. II. On completeness for $W[1]$, *Theoretical Computer Science*, 141(1-2):109-131, 1995.
- [11] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy, Interactive proofs and the hardness of approximating cliques, *Journal of the ACM (ACM)* 43 (2): 268-292, 1996.
- [12] Bin Fu, Separating PH from PP by relativization, *Acta Math. Sinica* 8(3):329-336, 1992.
- [13] V. Kabanets and R. Impagliazzo, Derandomizing polynomial identity tests means proving circuit lower bounds, *STOC*, pp. 355-364, 2003.
- [14] Adam Klivans and Rocco A. Servedio, Learning DNF in time $2^{\tilde{O}(n^{1/3})}$, *STOC*, pp. 258-265, 2001.

- [15] Ioannis Koutis, Faster algebraic algorithms for path and packing problems, Proceedings of the International Colloquium on Automata, Language and Programming (ICALP), LNCS, vol. 5125, Springer, pp. 575-586, 2008.
- [16] M. Minsky and S. Papert, Perceptrons (expanded edition 1988), MIT Press, 1968.
- [17] R. Motwani and P. Raghavan, Randomized Algorithms, Cambridge University Press, 1995.
- [18] Moni Naor, Leonard J. Schulman and Aravind Srinivasan, Splitters and near-optimal derandomization, FOCS, pp. 182-191, 1995.
- [19] Ran Raz and Amir Shpilka, Deterministic polynomial identity testing in non-commutative models, Computational Complexity 14(1): 1-19, 2005.
- [20] U. Schöning, A probabilistic algorithm for k -SAT based on limited local search and restart, Algorithmica, vol 32, pp. 615-623, 2002.
- [21] A. Shamir, $IP = PSPACE$, Journal of the ACM, 39(4): 869-877, 1992.
- [22] Ryan Williams, Finding paths of length k in $O^*(2^k)$ time, Information Processing Letters, 109, 315-318, 2009.